



**Barcelona  
Supercomputing  
Center**

*Centro Nacional de Supercomputación*

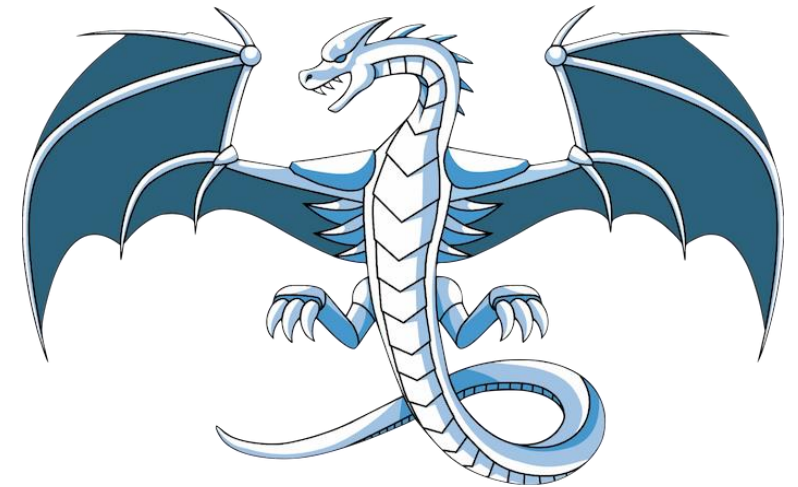
# SW Toolchain for RISC-V Vector Extensions

**Roger Ferrer Ibáñez**  
`roger.ferrer@bsc.es`

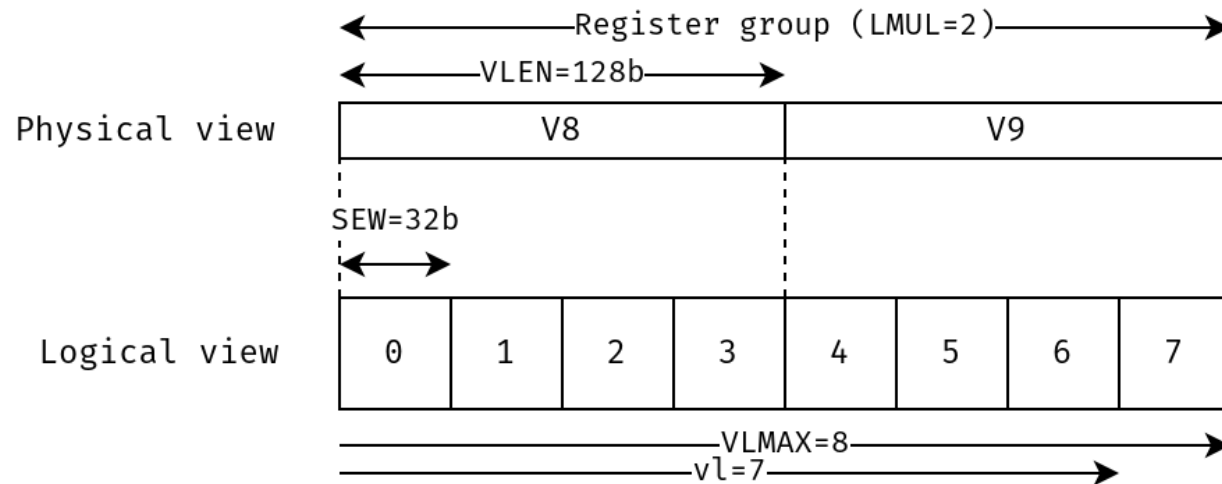
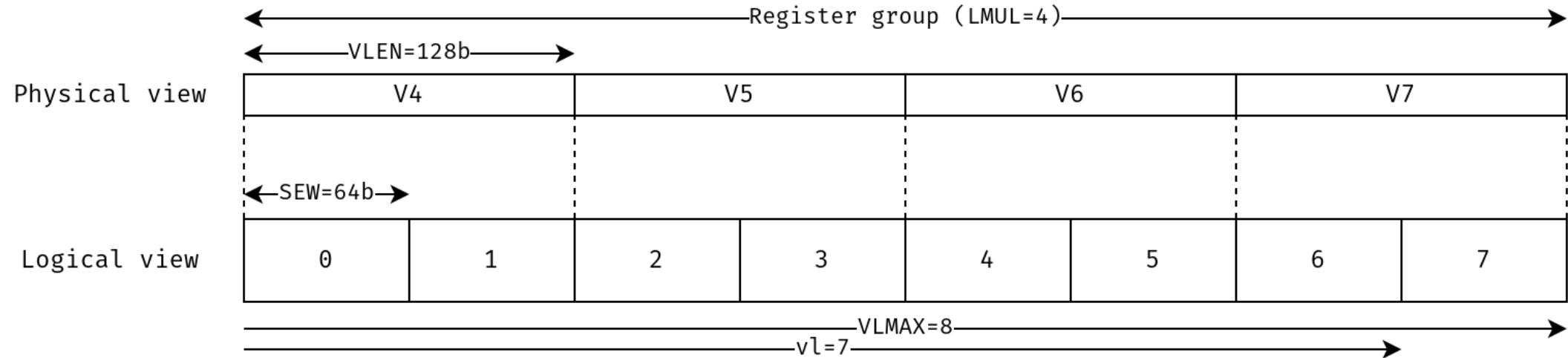
May 3rd 2022

Spring 2022 – RISC-V Week - Paris

# Acknowledgements



# RISC-V Vector Extension (RVV)



vtype = <SEW, LMUL, policy>



# Flexibility





# Challenges in code generation



**Barcelona  
Supercomputing  
Center**

*Centro Nacional de Supercomputación*

# RVV Architectural State and Instructions

%dc	= fadd <2 x double> %da, %db	→ vl=2, sew=64, lmul=1
%sc	= fadd <4 x float> %sa, %sb	→ vl=4, sew=32, lmul=1
%sc2	= fadd <8 x float> %sa2, %sb2	→ vl=8, sew=32, lmul=2
%sch	= fadd <2 x float> %sah, %sbh	→ vl=2, sew=32, lmul=1/2

↓  
vfadd.vv

# Current approach

`%sc = fadd <4 x float> %sa, %sb`

VLEN=128b



`%3:vr = nofpexcept PseudoVFADD_VV_M1 %0:vr, %1:vr, -1, 5, implicit $frm`

`vl=VLMAX`

`vtype=<sew=32,lmul=1>`

# What about setting the context

```
%3:vr = nofpexcept PseudoVFADD_VV_M1 %0:vr, %1:vr, -1, 5, implicit $frm
```



```
dead %4:gpr = PseudoVSETVLIX0 $x0, 80, implicit-def $vl, implicit-def $vtype
```

```
%3:vr = nofpexcept PseudoVFADD_VV_M1 %0:vr, %1:vr, -1, 5,  
                    implicit $frm, implicit $vl, implicit $vtype
```



# Challenges that impact the user of RVV



**Barcelona  
Supercomputing  
Center**

*Centro Nacional de Supercomputación*

# Intrinsics



# Vectorization



# LLVM and predication

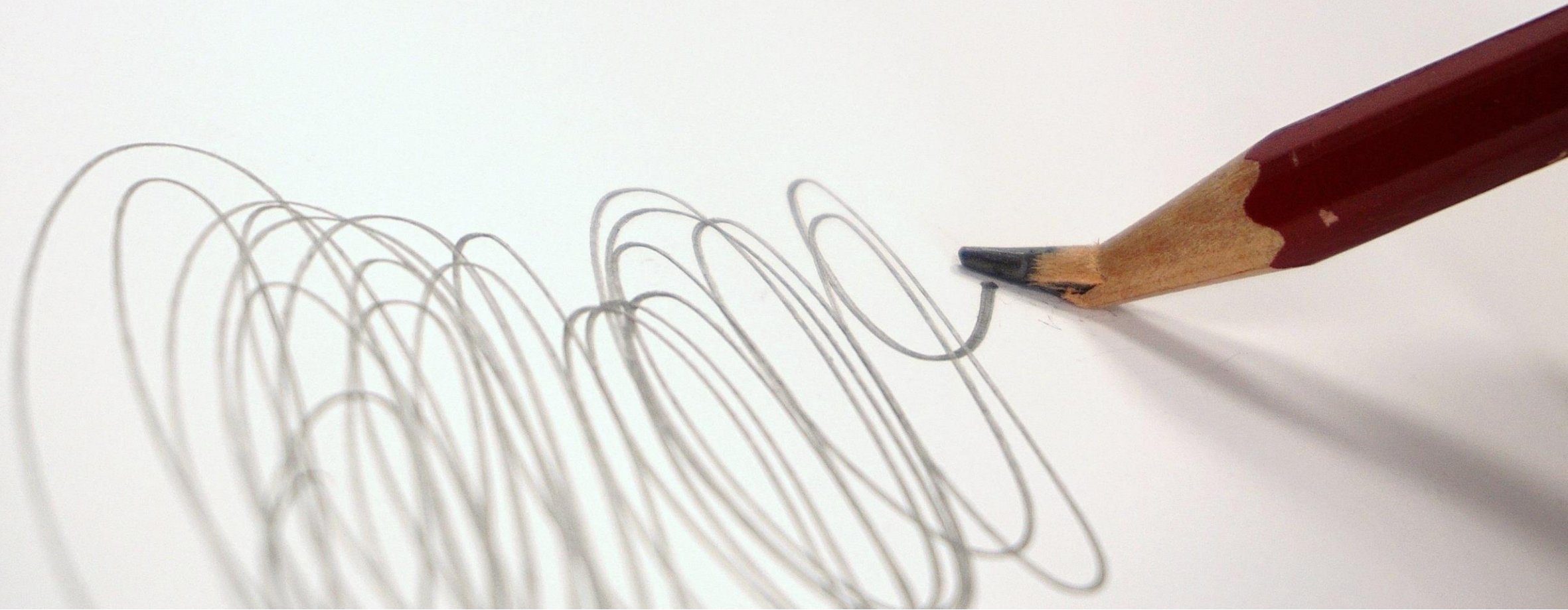


Image by [AxxLC](#) from [Pixabay](#)



# Vector Predication

Scalar operation (add two double precision values)

```
%sc = fadd double %sa, %sb
```

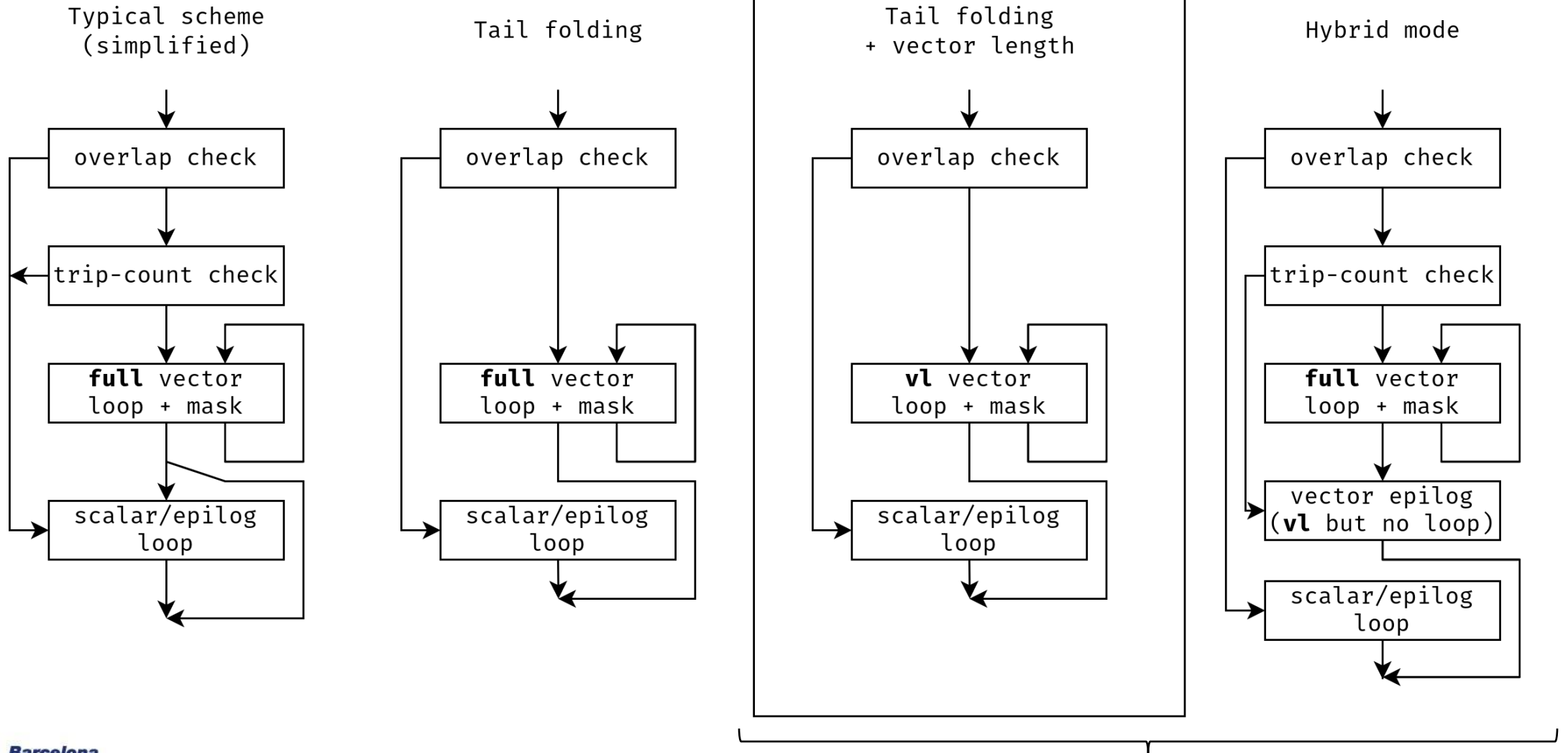
Element-wise extension to whole vectors (add two double precision vector values)

```
%vc = fadd <8 x double> %va, %vb  
%vla.c = fadd <vscale x 1 x double> %vla.a, %vla.b
```

Vector Predication allows us to specify mask and vector length operands

```
%vc = call <8 x double> @llvm.vp.fadd.nxv1f64(  
    <8 x double> %vla.a,  
    <8 x double> %vla.b,  
    <8 x i1> %mask, i32 %vl)  
  
%vla.c = call <vscale x 1 x double> @llvm.vp.fadd.nxv1f64(  
    <vscale x 1 x double> %vla.a,  
    <vscale x 1 x double> %vla.b,  
    <vscale x 1 x i1> %mask, i32 %vl)
```

# Loop Vectorisation at EPI



# Example DAXPY kernel

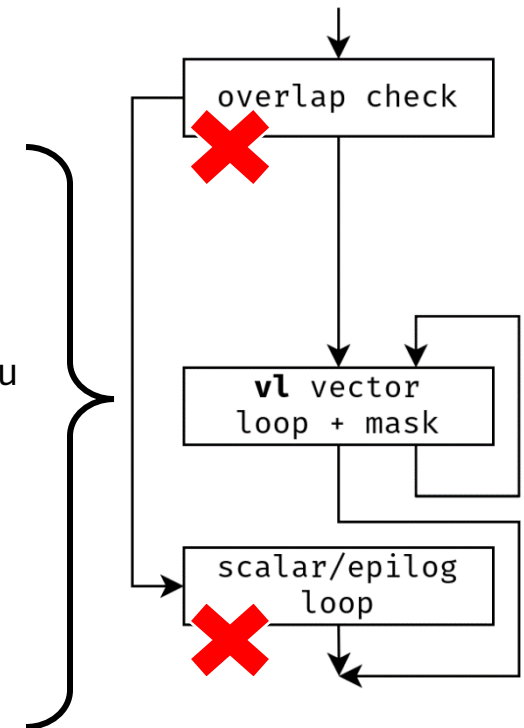
```
void daxpy(double a,  
          double * restrict dx,  
          double * restrict dy,  
          int n) {  
    for (int i = 0; i < n; i++) {  
        dy[i] += a * dx[i];  
    }  
}
```

You can try it at

<https://repo.hca.bsc.es/epic/z/iBdt4p>

```
daxpy:  
    blez    a2, .LBB0_3  
    li      a3, 0  
    slli    a2, a2, 32  
    srli    a6, a2, 32  
.LBB0_2:  
    slli    a4, a3, 3  
    add     a5, a0, a4  
    sub     a2, a6, a3  
    vsetvli a2, a2, e64, m1, ta, mu  
    vle64.v v8, (a5)  
    add     a4, a4, a1  
    vle64.v v9, (a4)  
    vfmac.vf v9, fa0, v8  
    add     a3, a3, a2  
    vse64.v v9, (a4)  
    bne     a3, a6, .LBB0_2  
.LBB0_3:  
    ret
```

Tail folding  
+ vector length





**Barcelona  
Supercomputing  
Center**

*Centro Nacional de Supercomputación*

# Thank you!

The European Processor Initiative (EPI) has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement EPI-SGA1: 826647 and under EPI-SGA2: 101036168. Please see <http://www.european-processor-initiative.eu> for more information.

The European PILOT project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No.101034126. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Spain, Italy, Switzerland, Germany, France, Greece, Sweden, Croatia and Turkey.

The MEEP project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 946002. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Spain, Croatia, Turkey

`roger.ferrer@bsc.es`